File Manage Service

Summary

This service provides the common functions for handling the attached files, which are required when handling the business logic, in order to use ***file list inquiry and file download*** functions. The ***file registration*** provides the client JavaScript and its sample.

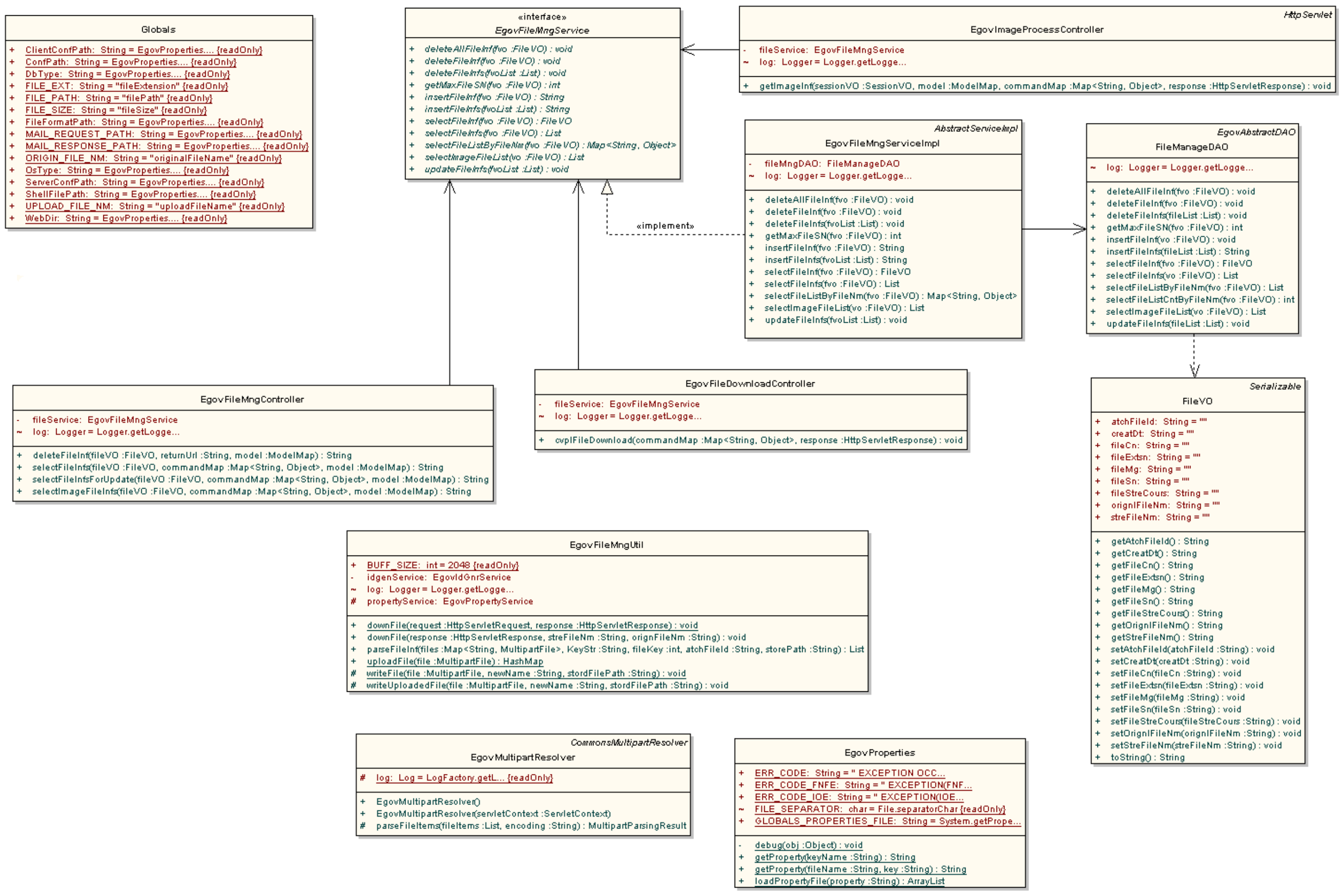
This function uses ***File Upload/Download*** function of the e- government standard framework execution environment.

Description

The common function for file upload is composed of ***EgovFileMngUtil*** class that parses files sent to the server by ***MultipartRequest*** and create and save data and ***EgovFileMngService*** service which manages file information. (refer to the related sources)

The part of JSP that handles file upload is based on Dynamic HTML and the script for it is included in ***EgovMultiFile.js***. Apply these to the screens that need upload, based on ***Manual***.

Class diagram



Globals
+ ClientConfPath: String = EgovProperties.... {readOnly}
+ ConfPath: String = EgovProperties.... {readOnly}
+ DbType: String = EgovProperties.... {readOnly}
+ FILE_EXT: String = "fileExtension" {readOnly}
+ FILE_PATH: String = "filePath" {readOnly}
+ FILE_SIZE: String = "fileSize" {readOnly}
+ FileFormatPath: String = EgovProperties.... {readOnly}
+ MAIL_REQUEST_PATH: String = EgovProperties.... {readOnly}
+ MAIL_RESPONSE_PATH: String = EgovProperties.... {readOnly}
+ ORIGIN_FILE_NM: String = "originalFileName" {readOnly}
+ OsType: String = EgovProperties.... {readOnly}
+ ServerConfPath: String = EgovProperties.... {readOnly}
+ ShellFilePath: String = EgovProperties.... {readOnly}
+ UPLOAD_FILE_NM: String = "uploadFileName" {readOnly}
+ WebDir: String = EgovProperties.... {readOnly}
«interface»
EgovFileMngService
+ deleteAllFileInf(fvo :FileVO) : void
+ deleteFileInf(fvo :FileVO) : void
+ deleteFileInfs(fvoList :List) : void
+ getMaxFileSN(fvo :FileVO) : int
+ insertFileInf(fvo :FileVO) : String
+ insertFileInfs(fvoList :List) : String
+ selectFileInf(fvo :FileVO) : FileVO
+ selectFileInfs(fvo :FileVO) : List
+ selectFileListByFileNm(fvo :FileVO) : Map<String, Object>
+ selectImageFileList(vo :FileVO) : List
+ updateFileInfs(fvoList :List) : void
HttpServlet
EgovImageProcessController
- fileService: EgovFileMngService
~ log: Logger = Logger.getLogge...
+ getImageInf(sessionVO :SessionVO, model :ModelMap, commandMap :Map<String, Object>, response :HttpServletResponse) : void
«implement»
AbstractServiceImpl
EgovFileMngServiceImpl
- fileMngDAO: FileManageDAO
~ log: Logger = Logger.getLogge...
+ deleteAllFileInf(fvo :FileVO) : void
+ deleteFileInf(fvo :FileVO) : void
+ deleteFileInfs(fvoList :List) : void
+ getMaxFileSN(fvo :FileVO) : int
+ insertFileInf(fvo :FileVO) : String
+ insertFileInfs(fvoList :List) : String
+ selectFileInf(fvo :FileVO) : FileVO
+ selectFileInfs(fvo :FileVO) : List
+ selectFileListByFileNm(fvo :FileVO) : Map<String, Object>
+ selectImageFileList(vo :FileVO) : List
+ updateFileInfs(fvoList :List) : void
EgovAbstractDAO
FileManageDAO
~ log: Logger = Logger.getLogge...
+ deleteAllFileInf(fvo :FileVO) : void
+ deleteFileInf(fvo :FileVO) : void
+ deleteFileInfs(fileList :List) : void
+ getMaxFileSN(fvo :FileVO) : int
+ insertFileInf(vo :FileVO) : void
+ insertFileInfs(fileList :List) : String
+ selectFileInf(fvo :FileVO) : FileVO
+ selectFileInfs(vo :FileVO) : List
+ selectFileListByFileNm(fvo :FileVO) : List
+ selectFileListCntByFileNm(fvo :FileVO) : int
+ selectImageFileList(vo :FileVO) : List
+ updateFileInfs(fileList :List) : void
EgovFileMngController
- fileService: EgovFileMngService
~ log: Logger = Logger.getLogge...
+ deleteFileInf(fileVO :FileVO, returnUrl :String, model :ModelMap) : String
+ selectFileInfs(fileVO :FileVO, commandMap :Map<String, Object>, model :ModelMap) : String
+ selectFileInfsForUpdate(fileVO :FileVO, commandMap :Map<String, Object>, model :ModelMap) : String
+ selectImageFileInfs(fileVO :FileVO, commandMap :Map<String, Object>, model :ModelMap) : String
EgovFileDownloadController
- fileService: EgovFileMngService
~ log: Logger = Logger.getLogge...
+ cvplFileDownload(commandMap :Map<String, Object>, response :HttpServletResponse) : void
Serializable
FileVO
+ atchFileId: String = ""
+ creatDt: String = ""
+ fileCn: String = ""
+ fileExtsn: String = ""
+ fileMg: String = ""
+ fileSn: String = ""
+ fileStreCours: String = ""
+ orignlFileNm: String = ""
+ streFileNm: String = ""
+ getAtchFileId() : String
+ getCreatDt() : String
+ getFileCn() : String
+ getFileExtsn() : String
+ getFileMg() : String
+ getFileSn() : String
+ getFileStreCours() : String
+ getOrignlFileNm() : String
+ getStreFileNm() : String
+ setAtchFileId(atchFileId :String) : void
+ setCreatDt(creatDt :String) : void
+ setFileCn(fileCn :String) : void
+ setFileExtsn(fileExtsn :String) : void
+ setFileMg(fileMg :String) : void
+ setFileSn(fileSn :String) : void
+ setFileStreCours(fileStreCours :String) : void
+ setOrignlFileNm(orignlFileNm :String) : void
+ setStreFileNm(streFileNm :String) : void
+ toString() : String
EgovFileMngUtil
+ BUFF_SIZE: int = 2048 {readOnly}
- idgenService: EgovIdGnrService
~ log: Logger = Logger.getLogge...
# propertyService: EgovPropertyService
+ downFile(request :HttpServletRequest, response :HttpServletResponse) : void
+ downFile(response :HttpServletResponse, streFileNm :String, orignFileNm :String) : void
+ parseFileInf(files :Map<String, MultipartFile>, KeyStr :String, fileKey :int, atchFileId :String, storePath :String) : List
+ uploadFile(file :MultipartFile) : HashMap
# writeFile(file :MultipartFile, newName :String, stordFilePath :String) : void
# writeUploadedFile(file :MultipartFile, newName :String, stordFilePath :String) : void
CommonsMultipartResolver
EgovMultipartResolver
# log: Log = LogFactory.getL... {readOnly}
+ EgovMultipartResolver()
+ EgovMultipartResolver(servletContext :ServletContext)
# parseFileItems(fileItems :List, encoding :String) : MultipartParsingResult
EgovProperties
+ ERR_CODE: String = " EXCEPTION OCC...
+ ERR_CODE_FNFE: String = " EXCEPTION(FNF...
+ ERR_CODE_IOE: String = " EXCEPTION(IOE...
~ FILE_SEPARATOR: char = File.separatorChar {readOnly}
+ GLOBALS_PROPERTIES_FILE: String = System.getPrope...
- debug(obj :Object) : void
+ getProperty(keyName :String) : String
+ getProperty(fileName :String, key :String) : String
+ loadPropertyFile(property :String) : ArrayList

Related sources

Type	***Target source name***	***Notes***
Controller	egovframework.com.cmm.web.EgovFileMngController.java	Controller class for file upload and inquiry
Controller	egovframework.com.cmm.web.EgovFileDownloadController.java	Controller class for file download

Service	egovframework.com.cmm.service.EgovFileMngService.java	Service interface for file management
ServiceImpl	egovframework.com.cmm.service.impl.EgovFileMngServiceImpl.java	Service implementation class for file management
VO	egovframework.com.cmm.service.FileVO.java	VO class for file management
DAO	egovframework.com.cmm.service.imp.FileManageDAO.java	Data processing class for file information management
Component	egovframework.com.cmm.service.EgovFileMngUtil.java	Util class for creating file and storing file information in DB
JSP	/WEB- INF/jsp/egovframework/cmm/fms/EgovFileList.jsp	File list inquiry and list inquiry page for download processing
js	/js/egovframework/cmm/fms/EgovMultiFile.js	Javascript file for file upload processing

Related table

Table name	***Table name (English)***	***Note***
File information	COMTNFILE	Manage file header information
Detailed file information	COMTNFILEDETAIL	Mange detailed file information

Configuration

The followings are the environment configuration required to enable the file upload function.

context- common.xml

```
<!-- MULTIPART RESOLVERS -->
<!-- regular spring resolver -->
<bean id="spring.RegularCommonsMultipartResolver"
  class="org.springframework.web.multipart.commons.CommonsMultipartResolver">
     <property name="maxUploadSize" value="100000000" />
     <property name="maxInMemorySize" value="100000000" />
</bean>

<!-- custom multi file resolver -->
<bean id="local.MultiCommonsMultipartResolver"
   class="egovframework.com.cmm.web.EgovMultipartResolver">
     <property name="maxUploadSize" value="100000000" />
     <property name="maxInMemorySize" value="100000000" />
</bean>
```

Since this common service provides multiple file upload, register egovframework.com.cmm.web.EgovMultipartResolver class as MultiCommonsMultipartResolver and, if required, modify the value of maxUploadSize, maxInMemorySize to specify the maximum uploadable file size. ※ Add spring.RegularCommonsMultipartResolver as Default value.

context- properties.xml

```
<bean name="propertiesService" class="egovframework.rte.fdl.property.impl.EgovPropertyServiceImpl"
   destroy- method="destroy">
   <property name="properties">
      <map>
         <entry key="pageUnit" value="10"/>
```

```
            <entry key="pageSize" value="10"/>
            <entry key="Globals.fileStorePath" value="/product/jeus/egovProps/upload/"/>
         </map>
      </property>
  </bean>
```

The value of Globals.fileStorePath specifies the physical location where the file is stored actually and it should express File.seperator, at the end of the address, which is consistent with the OS.

context- idgen.xml

```
            <bean name="egovFileIdGnrService"
                        class="egovframework.rte.fdl.idgnr.impl.EgovTableIdGnrService"
                        destroy- method="destroy">
                        <property name="dataSource" ref="dataSource" />
                        <property name="strategy" ref="fileStrategy" />
                        <property name="blockSize"  value="10"/>
                        <property name="table"                  value="COMTECOPSEQ"/>
                        <property name="tableName"value="FILE_ID"/>
            </bean>
            <bean name="fileStrategy"
                        class="egovframework.rte.fdl.idgnr.impl.strategy.EgovIdGnrStrategyImpl">
                        <property name="prefix" value="FILE_" />
                        <property name="cipers" value="15" />
                        <property name="fillChar" value="0" />
            </bean>
```

- In order to use ID Generation Service, add FILE_ID to COMTECOPSEQ, the sequence storing table.

```
  CREATE TABLE COMTECOPSEQ ( table_name varchar(16) NOT NULL,
                             next_id DECIMAL(30) NOT NULL,
                             PRIMARY KEY (table_name));
  INSERT INTO COMTECOPSEQ VALUES('FILE_ID','0');
```

Manual

When the above configuration is completed, carry out the following four additional coding tasks, if appropriate, based on the business logic.

The Javascript used in EgovFileList.jsp to inquire and delete files ***uses Form name as fim***. It is recommended to comply with the Form name to reduce additional works.

File Upload (first registration)

JSP/Script (File Upload)

To upload a file, the firm used on the screen should comply with the following style.

```
  <form name="frm" method="post" enctype="multipart/form- data" >
   <input type="hidden" name="posblAtchFileNumber" value="maximum registerable number of files" />
   .....
  </form>
```

Register the file on the screen so that you can use EgovMultiFile.js.

```
  <script type="text/javascript" src="<c:url value='/js/egovframework/cmm/fms/EgovMultiFile.js'/>" ></script>
```

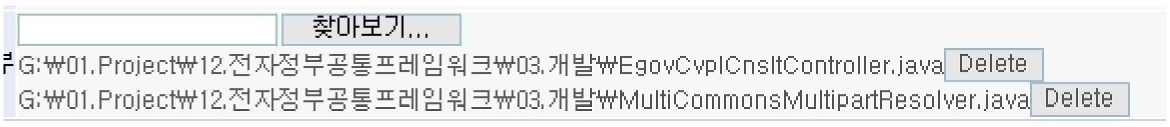
In order to be able to specify the location what file objects the script will use and where the script will stack them, insert the following HTML code and script

```
  <table width="680px" cellspacing="0" cellpadding="0" border="0" align="center" class="UseTable">
   <tr>
    <td><input name="file_1" id="egovComFileUploader" type="file" /></td>
   </tr>
   <tr>
    <td>
     <div id="egovComFileList"></div>
    </td>
   </tr>
  </table>
```

Add them at the end of the HTML code. In the following sample code, the value of document.frm.posblAtchFileNumber.value is used to contain the maximum uploadable number of files and it should be added, though the field is not used. If there is no value, there are three default values and changeable if required.

```
  <script type="text/javascript">
   var maxFileNum = document.frm.posblAtchFileNumber.value;
   if(maxFileNum==null || maxFileNum==""){
     maxFileNum = 3;
    }
   var multi_selector = new MultiSelector( document.getElementById( 'egovComFileList' ), maxFileNum );
   multi_selector.addElement( document.getElementById( 'egovComFileUploader' ) );
  </script>
```

If you write the script and HTML code and then call the screen, you can view the following screen. If the number of files is reached the maximum, the Find File button is enabled. Pushing the Delete button will delete the file object.



```
G:₩01.Project₩12.전자정부공통프레임워크₩03.개발₩EgovCvplCnsltController.java Delete
G:₩01.Project₩12.전자정부공통프레임워크₩03.개발₩MultiCommonsMultipartResolver.java Delete
```

Controller (File Upload)

In the controller for business, add the following to receive MultipartHttpServletRequest and handle the file request.

```
import egovframework.com.cmm.service.EgovFileMngService;
import egovframework.com.cmm.service.EgovFileMngUtil;

@Resource(name="EgovFileMngService")
private EgovFileMngService fileMngService;

@Resource(name="EgovFileMngUtil")
private EgovFileMngUtil fileUtil;
```

```
   public String insertBoardArticle(
              final MultipartHttpServletRequest multiRequest,
              @ModelAttribute("searchVO") BoardVO boardVO,
              @ModelAttribute("board") Board board,
              SessionVO sessionVO,
              ModelMap model,
              SessionStatus status) throws Exception{

              List<FileVO> _result = null;
              String _atchFileId = "";
              final Map<String, MultipartFile> files = multiRequest.getFileMap();
              if(!files.isEmpty()){
                _result = fileUtil.parseFileInf(files, "BBS_", 0, "", "");
                _atchFileId = fileMngService.insertFileInfs(_result);   //When the file is created, return the created
attachment ID.
              }
     .....
 }
```

- The parseFileInf method of ***EgovFileMngUtil*** class has five arguments; file object, value, file serial number, file ID and storing path. When the file ID is an empty string or null, a new file ID is created by using ***IDGenerationService***.
- If the storing path is an empty string or null, the one specified in ***context- properties.xm*** is used. Select 0 for the file serial number when registering for the first time
- The value is used when storing an actual file at a physical location as a changed name and when creating a file name. The current file creation rule is 'distinct value"+"yyyyMMddhhmmssSSS type TimeStamp value"+"file key".

File Select

1. If any attached file is found in the business logic, send the attachment ID to Parameter as in the sample code.
2. When the file list inquiry is completed, the file name and size are displayed in the form of [new text document].txt[405byte].
3. . For File Select, the button for deleting files is not enabled but only the link for file download exists.

```
<c:import url="/cmm/fms/selectFileInfs.do" >
  <c:param name="param_atchFileId" value="${result.atchFileId}" />
</c:import>
```

File Delete

1. If necessary to delete a attached file (for example, to modify a post), send the registered attachment ID to Parameter as in File Select.
2. On the contrary to File Select, there is no download link and the button for deleting files is enabled.

```
<c:import url="/cmm/fms/selectFileInfsForUpdate.do" >
  <c:param name="param_atchFileId" value="${result.atchFileId}" />
</c:import>
```

- When a file is deleted, Submit() occurs for the entire screen. Designate the return page after deleting.

- To do so, add the following HTML object to the business function page to be deleted and designate the return page as in the following.

```
<code html>
 <form name="frm" method="post">
   <input type="hidden" name="returnUrl" value="<c:url value='/cop/bbs/forUpdateBoardArticle.do'/>"/>
 </form>
</code>
```

File Upload(modify and register)

To upload a file, the form used on the screen should comply with the following.

```
  <form name="frm" method="post" enctype="multipart/form- data" >
    <input type="hidden" name="posblAtchFileNumber" value="max registerable number of files" />
    .....
  </form>
```

JSP/Script (File Upload – modify & register)

This is for attaching or delete a file while modification. In the current module, the file deleting and storing are separate and they are treated as different ***Transaction***. You have to register EgovMultiFile.js first so that you can use it similar to when you register it for the first time.

```
  <script type="text/javascript" src="<c:url value='/js/egovframework/cmm/fms/EgovMultiFile.js'/>" ></script>
```

As above mentioned, while modifying, the existing registered file is deleted or newly registered. Therefore the part of entering a file newly and the part of deleting a file should be displayed on a screen. The file list inquiry and deleting is handled in the same way of deleting an existing file. For stacking files, prepare the following HTML code.

```
 <div id="file_upload_posbl"   style="display:none;" >
  <table width="680px" cellspacing="0" cellpadding="0" border="0" align="center" class="UseTable">
    <tr>
      <td><input name="file_1" id="egovComFileUploader" type="file" /></td>
    </tr>
    <tr>
      <td>
          <div id="egovComFileList"></div>
      </td>
    </tr>
  </table>
 </div>
 <div id="file_upload_imposbl"   style="display:none;" >
    <table width="680px" cellspacing="0" cellpadding="0" border="0" align="center" class="UseTable">
     <tr>
      <td><spring:message code="common.imposbl.fileupload" /></td>
     </tr>
    </table>
 </div>
```

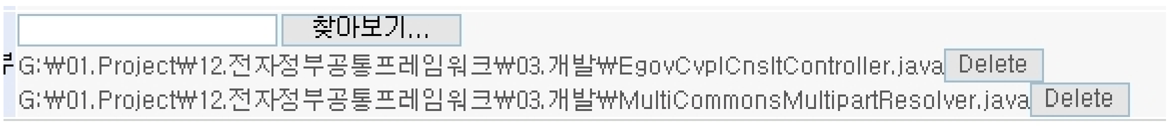
The reason that DIV is divided into file_upload_posbl and file_upload_imposbl is for message handling and need some improvement.

Add the script at the end of HTML code.   In the following sample code, the value of document.frm.posblAtchFileNumber.value is used to contain the maximum uploadable number of files and it should be added, though the field is not used. The Form name used for file upload is frm, similar to the file registeration.

```
   <script type="text/javascript">

   var existFileNum = document.frm.fileListCnt.value; // The value is contained in the part of file list inquiry.
   var maxFileNum = document.frm.posblAtchFileNumber.value;
   // Set the max attachable number of files to the form valve
   var uploadableFileNum = maxFileNum -  existFileNum; // Substract the already registered number of files
from the max attachable number of files .
   if(uploadableFileNum<0) {
      uploadableFileNum = 0;
   }
   if(uploadableFileNum != 0){
      fn_egov_check_file('Y');
      var multi_selector = new MultiSelector( document.getElementById( 'egovComFileList' ),
uploadableFileNum );
      multi_selector.addElement( document.getElementById( 'egovComFileUploader' ) );
   }else{
      fn_egov_check_file('N');
   }
   </script>
```

If you write the script and HTML code and then call the screen, you can view the following screen. If the max number of files is registered, the Find File button is enabled. Pushing the Delete button will delete the file object.



G:₩01.Project₩12.전자정부공통프레임워크₩03.개발₩EgovCvplCnsltController.java Delete
G:₩01.Project₩12.전자정부공통프레임워크₩03.개발₩MultiCommonsMultipartResolver.java Delete

Controller (File Upload – Modify & Register)

```
import egovframework.com.cmm.service.EgovFileMngService;
import egovframework.com.cmm.service.EgovFileMngUtil;

@Resource(name="EgovFileMngService")
private EgovFileMngService fileMngService;

@Resource(name="EgovFileMngUtil")
private EgovFileMngUtil fileUtil;

public String updateBoardArticle(
final MultipartHttpServletRequest multiRequest,
@ModelAttribute("searchVO") BoardVO boardVO,
@ModelAttribute("board") Board board,
SessionVO sessionVO,
ModelMap model,
SessionStatus status) throws Exception{

String _atchFileId = boardVO.getAtchFileId();/ Call the file ID of the function that is modified based on the
business function.
final Map<String, MultipartFile> files = multiRequest.getFileMap();
if(!files.isEmpty()){
         if("".equals(_atchFileId)){
                  List<FileVO> _result = fileUtil.parseFileInf(files, "BBS_", 0, _atchFileId, "");
```

```
                        _atchFileId = fileMngService.insertFileInfs(_result); // No attachment ID.
                        board.setAtchFileId(_atchFileId); // Set up the attachment ID created by the relevant
business rule.
            }else{
                        FileVO fvo = new FileVO();
                        fvo.setAtchFileId(_atchFileId); // To obtain the final file serial number, set up the current
attachment ID in VO.
                        int _cnt = fileMngService.getMaxFileSN(fvo); // Obtain the final file serial number that
belongs to the attachment ID.
                        List<FileVO> _result = fileUtil.parseFileInf(files, "BBS_", _cnt, _atchFileId, "");
                        fileMngService.updateFileInfs(_result);
            }
}
......
```

- The basic flow is the same as the first creation. If a file ID exist already, obtain the final FILE_SN(file serial number) that belongs to the attachment ID.
- The file serial number and attachment ID should be provided as arguments to parseFileInf method. The file creation logic after it is the same as the first creation.
- If a file exist already (updateFileInfs method), ensure that there is no return value.

References

- Refer to the execution environment: File Upload/Download